A Conceptual Graph Model for W3C Resource Description Framework

Olivier Corby, Rose Dieng, Cédric Hébert

A Conceptual Graph Model for W3C Resource Description Framework

Olivier Corby, Rose Dieng, and Cédric Hébert

INRIA Sophia Antipolis
{Olivier.Corby,Rose.Dieng}@inria.fr,
http://www.inria.fr/acacia
Acacia Project
2004 route des Lucioles, BP 93
06902 Sophia Antipolis Cedex
France

Abstract. With the aim of building a "Semantic Web", the content of the documents must be explicitly represented through metadata in order to enable contents-guided search. Our approach is to exploit a standard language (RDF, recommended by W3C) for expressing such metadata and to interpret these metadata in conceptual graphs (CG) in order to exploit querying and inferencing capabilities enabled by CG formalism. The paper presents our mapping of RDF into CG and its interest in the context of the semantic Web[1].

1 Introduction

The Web is recognized as a fabulous information repository, with millions of heterogeneous information sources available throughout the world. But the existing keyword-based search engines do not take into account the semantics of the documents accessible through the Web. The user can be easily overwhelmed by the huge number of answers (not always relevant) to a query. Therefore, the need of a "Semantic Web" is more and more emphasized [2, 3]. The semantics of the documents must be explicitly represented through semantic metadata in order to enable semantic-contents-guided search [1]. Several proposals have been offered to this end, for example Ontobroker [8] and Shoe [12], that rely on extensions of HTML and exploitation of ontologies. In [9], the authors analyse several languages that may be used for representing metadata. They notice the following problems to be solved when dealing with large amounts of semi-structured information: searching information, extracting information, maintaining weakly structured sources, generating documents. They emphasize the importance of relying on standards that are widely accepted by the Web community. [9] stresses that knowledge representation languages offered in artificial intelligence (AI) seem attractive for this aim, but suffer from the lack of wide acceptance. We

[1] Published in the proceedings of the International Conference on Conceptual Structures, ICCS 2000, Darmstadt, August 2000. Springer Verlag.

are convinced of the interest of AI representation languages that enable not only the representation of metadata but also support inferences on them. Among such AI knowledge representation formalisms, [13, 14] stress the advantages of conceptual graph (CG) formalism for expressing metadata. Another approach is to exploit a standard language for expressing metadata and to be able to interpret these metadata in conceptual graphs in order to exploit querying and inferencing capabilities enabled by conceptual graph formalism.

RDF (Resource Description Framework) has been introduced and recommended by the World Wide Web Consortium (W3C) to enable descriptions of semantic metadata for the Semantic Web. RDF enables the addition of semantic information to a Web document, without making any assumptions about the structure of this document. The future will reveal whether RDF will be accepted as a standard for content descriptions of Web resources and whether it will be widely used by the authors of Web documents. In the event RDF is broadly accepted, the approach of automatically interpreting metadata expressed in RDF into conceptual graphs seems interesting: it will enable us both to rely on a standard and draw benefit from the advantages of conceptual graphs.

The purpose of the paper is to show that conceptual graphs can be used as a means to exploit RDF metadata to handle metadata-based search queries. After a description of the principles of RDF, we will detail the mapping of RDF and CG. Then, we discuss the interest of this approach in comparison to related work.

2 RDF

RDF is based on an underlying model with triples made of resource, property, and value.

- A resource is an entity accessible by an URI on the Web (e.g. an HTML or XML document). Resources are the elements described by RDF statements.
- A property defines a binary relation between resources and/or atomic values. A property enables us to attach information to resources, and provide descriptions for resources.
- A value can be either a simple character string or a resource. Reification enables one to transform a triple into a resource. The notion of collections permits us to define groups to which some properties are applied.

An RDF statement specifies a value for a property of a resource.

RDF has an XML syntax and can be seen as an object-oriented formalism for metadata statements. These metadata can rely on common ontologies represented using RDF Schema (RDFS).

RDF statements can be considered as triples (resource, property, value). The vocabulary used in these triples can be defined using RDFS, by a hierarchy of classes and a hierarchy of properties.

Contrary to object-oriented or frame-based representations, RDF relies on a property-centric approach. Anyone can define properties about Web resources,

in order to offer descriptions for these resources. In RDFS, properties are defined globally and not encapsulated in class definitions. They can be specialized using the subPropertyOf relationship. RDF/S offers three core classes:

- Resource (i.e. class of all objects),
- Property (i.e. class of all properties),
- Class (i.e. class of all classes).

Two core properties are provided: type and subClassOf. The classes can be specialized through the subClassOf relationship. A resource is said to be an instance of a given class by means of the type property. The range and domain core properties are used to define the range (resp. domain) of properties.

3 Mapping RDF to CG

The model of CG formalism [17, 6] is based on (1) a support made of a concept type lattice and of a relation type set possibly organized in hierarchy, a set of individual markers enabling the designation of instances, a conformity relation between markers and types, and (2) a base of conceptual graphs built on this support.

It therefore seems natural to translate a) the RDF statements into a base of CG-facts b) the hierarchy of classes appearing in an RDF schema into a concept type hierarchy in CG, and c) the hierarchy of properties appearing in a RDF schema into a relation type hierarchy in CG. Therefore we will rely on a CG model enabling us to build a relation type hierarchy.

3.1 Mapping of Basic RDF

A basic RDF statement says something like : 'the author of the resource found at `http://www.bookstore.org/id1971` is `John Rawls`'. It can be stated as a triple by this way :

```
author(http://www.bookstore.org/id1971, 'John Rawls')
```

Several statements can be written about the same resource, for example :

```
title(http://www.bookstore.org/id1971, 'A Theory of Justice')
date(http://www.bookstore.org/id1971, '1971')
```

Written with the RDF/XML syntax :

```
<rdf:Description about='http://www.bookstore.org/id1971'>
    <author>John Rawls</author>
    <title>A theory of Justice</title>
    <date>1971</date>
</rdf:Description>
```

This can be interpreted in CG as :

```
[Resource : http://www.bookstore.org/id1971] - {
    -> (author) -> [Literal : John Rawls]
    -> (title) -> [Literal : A theory of Justice]
    -> (date) -> [Literal : 1971]}
```

The principle of the mapping relies on considering an RDF description as an instance of a Resource CG concept type and the associated properties as relations of this concept. The designator of a Resource concept is the URI of the resource itself.

3.2 Mapping of Nested RDF Descriptions

The value of a basic RDF triple can also be an RDF description. For example, we can express that the value of the `bio` property is itself a description of another resource :

```
<rdf:Description about='http://www.bookstore.org/id1971'>
    <title>A theory of Justice</title>
    <author>John Rawls</author>
    <bio>
        <rdf:Description about='http://www.bookstore.org/John.Rawls'>
             <position>Philosopher</position>
        </rdf:Description>
    </bio>
</rdf:Description>
```

This RDF description will be translated into the following CG:

```
[Resource : http://www.bookstore.org/id1971] - {
   -> (title) -> [Literal : A theory of Justice]
   -> (author) -> [Literal : John Rawls]
   -> (bio) -> [Resource : http://www.bookstore.org/John.Rawls] ->
                    (position) -> [Literal : Philosopher]}
```

In case of nested RDF descriptions, the mapping consists of creating a Resource-typed concept for each nested resource description. Each nested resource concept is then linked to its embedded resource via a relation. For example, in the previous example, the nested resource `http://www.bookstore.org/John.Rawls` is linked by means of a `bio` relation to the embedding resource `http://www.bookstore.org/id1971`.

3.3 RDF Schema

RDF descriptions can be typed according to a predefined ontology called a RDF schema. Thus RDFS formalism enables provision of a vocabulary used for the RDF annotations. For example, the previous description can be typed as a description of a book, the class Book being itself defined in an RDF schema.

```
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
         xmlns:ns='http://www.inria.fr/acacia/iccs#'>
    <ns:Book rdf:about='http://www.bookstore.org/id1971'>
        <ns:author>John Rawls</ns:author>
        <ns:title>A theory of Justice</ns:title>
    </ns:Book>
</rdf:RDF>
```

The leading RDF markup with a `xmlns:rdf` attribute defines the RDF namespace `rdf` as a shortcut for the RDF URI, namely : `http://www.w3.org/-1999/02/22-rdf-syntax-ns#`. Each RDF markup must be prefixed with this namespace in order to identify what is RDF-related and what is application-specific. An XML namespace is also associated with the application schema, (say `ns`), in order to identify the markup correctly.

The corresponding CG is the following:

```
[Book : http://www.bookstore.org/id1971] - {
    -> (author) -> [Literal : John Rawls]
    -> (title) -> [Literal : A theory of Justice]}
```

Accordingly, when the RDF description is typed by a specific RDF Schema class, a concept of the corresponding type is created. In the example, the Resource is of type Book, hence a Book concept is created.

Concept and relation type names must be prefixed in a unique way, with a URI, to prevent name clashes from different schemas. For example, the concept type `Book` should be named : `http://www.inria.fr/acacia/iccs#Book` and the relation type `author` : `http://www.inria.fr/acacia/iccs#author`. For the sake of readability, we skip these prefixes for CG in the paper, but they are mandatory in the implementation.

4 Mapping RDF Schema

The RDF Schema, according to the current W3C Candidate Recommendation from March 2000 [5], allows the definition of classes and properties. Classes and properties can be refined in subclasses and subproperties. In RDF, properties are first class objects which exist by themselves. Hence, new properties can be added to existing classes in order to enable reuse of these classes.

4.1 Mapping of Classes

RDFS classes can be modelled as CG concept types. In order to map the Resource core class of RDFS, we introduce a Resource concept type at the top level of the CG concept type hierarchy.

```
concept type Resource
```

A RDFS class without any superclass explicitly indicated will be modelled by a subtype of Resource in the CG concept type hierarchy.

```
<rdfs:Class rdf:ID='Cl'/>
```

This can be translated into,

```
concept type Cl < Resource
```

For example :

```
<rdfs:Class rdf:ID='Book'/>
```

can be modelled as :

```
concept type Book < Resource
```

4.2 Mapping Subclasses

The subClassOf relation between classes in RDFS corresponds to the subtype relation (denoted ¡) between concept types in CG formalism. If an RDFS class Cl2 is defined as a subclass of Cl1, it will be modelled by a Cl2 concept type, subtype of the Cl1 concept type in the CG concept type hierarchy.

```
<rdfs:Class rdf:ID='Cl2'>
     <rdfs:subClassOf rdf:resource='#Cl1'/>
</rdfs:Class>
```

This can be translated into :

```
concept type Cl2 < Cl1
```

The Novel subclass of Book :

```
<rdfs:Class rdf:ID='Novel'>
     <rdfs:subClassOf rdf:resource='#Book'/>
</rdfs:Class>
```

can be modelled as a subtype of Book:

```
concept type Novel < Book
```

4.3 Mapping of Properties

A property is defined according to a domain (i.e. a class) and has an associated range that can be a literal or a class. For example, the title property can be defined with 'Book' domain and 'Literal' range. 'Literal' is prefixed with the RDFS namespace.

```
<rdf:Property ID='title'>
     <rdfs:domain rdf:resource='#Book'/>
     <rdfs:range rdf:resource='http://www.w3.org/TR/1999/PR-rdf-
         schema-19990303#Literal'/>
</rdf:Property>
```

A property definition can be modelled as a CG binary relation type with an associated signature that maps the domain and range to the related concept types. For instance, the previous example is translated into:

```
relation type title (Book, Literal)
```

4.4 Mapping of SubProperties

In RDFS, a subproperty can refine an existing property, in the same way as a subclass refines a class. It means that if p2 is a subproperty of p1, then :

```
p2(URI, v) => p1(URI, v)
```

that is, if a URI has value v for property p2, then it also has v as value for property p1.

For example, we define here the author property and its jointAuthor subproperty :

```
<rdf:Property ID='author'>
    <rdfs:domain rdf:resource='#Book'/>
    <rdfs:range rdf:resource='http://www.w3.org/TR/1999/PR-rdf-
          schema-19990303#Literal'/>
</rdf:Property>

<rdf:Property ID='jointAuthor'>
        <rdfs:subPropertyOf rdf:resource='#author'/>
</rdf:Property>
```

In terms of CGs, an RDF subproperty is translated into a relation type that is a subtype of the relation type translating the superproperty. In the example above, the jointAuthor relation type will be defined as a subtype of the author relation type.

```
relation type author (Book, Literal)
relation type jointAuthor < author
```

4.5 Remark: Limits of this mapping

Let us notice a limitation in the mapping from RDF to CG :

In the RDF Schema, a property can be defined with several classes as domain. However, this is not possible directly in CG because a relation type has only one signature. If this happens, three possibilities are offered to solve the problem :

1. If the domain classes have a common RDF Schema class ancestor, set the relation domain to that ancestor,
2. or else, define an abstract common superclass to all domain classes of the property and assign this new class as the domain of the relation signature. This is only possible if one masters the RDF Schema(s).
3. otherwise, set the domain of the property to Resource, the top level CG concept type that matches all resources. The property will then be allowed for all resources.

For example :

```
<rdf:Property ID='title'>
    <rdfs:domain rdf:resource='#Book'/>
    <rdfs:domain rdf:resource='#Show'/>
    <rdfs:range rdf:resource='http://www.w3.org/TR/1999/PR-rdf-
        schema-19990303#Literal'/>
</rdf:Property>
```

Solution 2 would consist of defining an abstract class, called 'Work' for example, having 'Book' and 'Show' as subclasses and, then, define the title relation type with the following signature :

```
concept type Work
concept type Book < Work
concept type Show < Work

relation type title (Work, Literal)
```

Otherwise, solution 3 would consist of defining title as :

```
relation type title (Resource, Literal)
```

4.6 Implementing the RDF Schema metamodel

The RDF metamodel is itself described in RDF, thus enabling the extension of the metamodel, by refining the predefined classes and properties.

For example, in RDF Schema, it is possible to define a 'Concept' metaclass that refines the 'Class' metaclass, and then define a schema in terms of 'Concept' instead of 'Class' :

```
<rdfs:Class rdf:ID="Concept">
    <rdfs:subClassOf="http://www.w3.org/TR/1999/PR-rdf-
          schema-19990303#Class/>
</rdfs:Class>

<ns:Concept rdf:ID="Paper"/>
```

We have implemented the whole RDF and RDF Schema metamodel in the CG support and enable metamodel extension.

5 More advanced features

5.1 Reification

A reified statement is an RDF statement about an RDF statement, as for example :

John K. Galbraith says :
'The author of the resource http://www.bookstore.org/id1971 is John Rawls'.

In RDF, the treatment of such a reified statement is somehow cumbersome and introduces specialized properties as shown below :

```
<rdf:Description>
      <rdf:subject resource="http://www.bookstore.org/id1971"/>
      <rdf:predicate resource="#author"/>
      <rdf:object>John Rawls</rdf:object>
      <rdf:type resource="http://www.w3.org/1999/02/22-rdf-
           syntax-ns#Statement"/>
      <attributedTo>John K. Galbraith</attributedTo>
</rdf:Description>
```

In CG, this can be done simply with a context named Statement :

```
[Statement: [Book: http://www.bookstore.org/id1971] -> (author) ->
                  [Literal:John Rawls]
] -> (attributedTo) -> [Literal : John K. Galbraith]
```

Reified RDF statements, i.e. : descriptions that are instances of the Statement class, are translated into CG by means of a predefined context named Statement. The RDF description itself is translated into CG following the standard mapping.

5.2 Mapping of Containers

The model of basic RDF relies on triples with single values. In the case where the value of a property is a set of values, the W3C recommendation defines containers such as bags, sequences and alternatives in order to hold such values. For example here, the authors are given as a set of names :

```
<rdf:Paper rdf:about='http://www.inria.fr/acacia/iccs2000'>
    <ns:authors>
        <rdf:Bag>
            <rdf:li>Cédric Hébert</rdf:li>
            <rdf:li>Olivier Corby</rdf:li>
            <rdf:li>Rose Dieng</rdf:li>
        </rdf:Bag>
    </ns:authors>
</rdf:Paper>
```

Containers can be handled by an adequate Bag concept which is related to its members by means of a member relation, called `rdf:li` :

```
[Paper:http://www.inria.fr/acacia/iccs2000] -> (authors) -> [Bag]-{
   -> (rdf:li) -> [Literal: Cédric Hébert]
   -> (rdf:li) -> [Literal: Olivier Corby]
   -> (rdf:li) -> [Literal: Rose Dieng]}
```

We introduce in the CG concept type hierarchy an abstract 'Container' concept type and three subtypes for the concrete containers (see the appendix on types at the end of the paper). We also introduce a `rdf:li` relation type :

```
    relation type rdf:li (Container, Resource)
```

5.3 Mapping Containers having aboutEach statements

RDF offers a means to factorize statements that apply to all members of a container. This is done with an aboutEach statement. In the example below, the bag is given the `auth` ID, and it is said that all members of the `auth` bag have INRIA as institute :

```
<rdf:Paper rdf:about='http://www.inria.fr/acacia/iccs2000'>
    <ns:authors>
        <rdf:Bag ID='auth'>
            <rdf:li>Cédric Hébert</rdf:li>
            <rdf:li>Olivier Corby</rdf:li>
            <rdf:li>Rose Dieng</rdf:li>
        </rdf:Bag>
    </ns:authors>
</rdf:Paper>

<rdf:Description aboutEach='#auth'>
        <ns:inst>INRIA</ns:inst>
</rdf:Description>
```

We can translate on the fly to distribute the factorized value to all bag members :

```
[Paper: http://www.inria.fr/acacia/iccs2000] -> (authors) -> [Bag]-
{
-> (rdf:li) -> [Literal:Cédric Hébert] -> (inst) -> [Literal:INRIA]
-> (rdf:li) -> [Literal:Olivier Corby] -> (inst) -> [Literal:INRIA]
-> (rdf:li) -> [Literal:Rose Dieng] -> (inst) -> [Literal:INRIA]}
```

6 Querying

The main interest of mapping RDF to CG is the adequacy between the two models, i.e. concepts and relations smoothly map onto classes and properties that are defined independently in CG as well as in RDF. Furthermore, it enables us to use RDF without any knowledge of Conceptual Graphs.

The second reason is the relevance of the CG projection operation to querying a RDF/CG base. Querying RDF metadata consists of retrieving RDF triples belonging to classes, taking specialization into account. This can be done through the projection operation.

Furthermore, thanks to the implementation platform that we have chosen, namely F. Southey's NOTIO [16], it is possible to parametrize precisely the graph matching process. Hence, it is possible to tune concept matching, including type and instance matching. For example, concepts may match according to (at least) one of the four conditions on concept types :

- first type is a supertype of the second
- first type is a subtype of the second
- first type is either a subtype or a supertype of the second
- concepts have same type.

Relation matching can also be parametrized, as well as other aspects of the graph matching. This functionality is well adapted to metadata information retrieval as it authorizes approximate matching along specialization and generalization and on relations and concepts.

In the current prototype, the query language is RDF itself. The user describes a partial RDF statement that he is looking for. The RDF query may hold variables, prefixed by '?', to indicate the parts that are unknown, the value of which should be returned by the query processor.

For example, let's look for books the author of which is John Rawls, and return their title :

```
<ns:Book rdf:about="?1">
    <ns:author>John Rawls</ns:author>
    <ns:title>?2</ns:title>
</ns:Book>
```

The RDF query is translated into the graph shown below :

```
[Book] - {
    -> (author) -> [Literal : John Rawls]
    -> (title) -> [Literal]}
```

The query processor projects the query graph on the CG base. The resulting (sub)graphs are translated back into RDF in order to be presented to the user in a uniform way.

The prototype also implements approximate search on literal values, thanks to the NOTIO matching scheme that enables us to attach customized match comparators to markers. We implemented such an approximate comparator that tests whether the query literal value is included into the graph literal value. Approximate query values are prefixed by the '∼' character.

It is then possible to send a query that searches an author, the value of which contains 'Rawls', as shown below.

```
<ns:Book rdf:about="?1">
    <ns:author>~Rawls</ns:author>
    <ns:title>?2</ns:title>
</ns:Book>
```

If several properties implement the *author* relationship, e.g. author, joint authors, etc. the problem may arise of taking all of them into account for query processing. In fact, RDF enables the association of a common external label to all these properties, say `author`. An advanced query GUI would be able to propose the abstract `author` property to the user and translate it to several CGs according to the signatures of the target author relation signatures (container, literal, etc.). In any case, this problem is relative to RDF, not to the CG translation model.

7 Implementation

We implemented a prototype using the NOTIO CG platform [16] and the VRP RDF parser from ICS Forth [19]. We exploited the possibility offered by NOTIO for parametrizing the projection (cf. generalization, specialization). Our prototype can translate classes and properties of RDF Schema and RDF statements, except the aboutEachPrefix RDF statement that is not presented here.

The translation of the RDF metadata into a base of CG-facts can be done automatically thanks to our prototype. Several ways of integrating such an automatic translator RDF-¿CG in a Web search engine can be thought out :

- A robot could access the Web documents and build a base of CG-facts corresponding to the translation into CG of their RDF metadata. The link between the CG-facts associated to a document and the document would be kept by the robot. This would be done before any requests from a user. Then, when a user makes a request to search a given document, this request would be translated into a CG-query. The parameters of the user's request (in particular, if the user wants to obtain approximate answers by enabling generalization or specialization) can also be exploited to parametrize the projection to be used. The results of the projection of the CG-query on the base of CG-facts constitute the answers to the user's request.

– Another possibility would be to let the search engine use the translator to build the CG dynamically, only after a request by the user.

The prototype currently runs as a Java servlet, accessed by means of a standard Internet navigator at a given URI. The user can type a query and send it to the system which performs the projection and sends back the answer in RDF. The resulting RDF statements are displayed by means of an XSLT stylesheet, thanks to James Clark's XT engine [7]. The URI contained in the resulting RDF statements are transformed into active HTML links on which the user may click. Hence, the whole process implements a conceptual search engine.

8 Conclusion and Discussion

Our approach delivers a more powerful and relevant search with the CG projection. In particular, the parametrization of the projection enables several levels of search.

Our approach takes advantage of the CG formalism, as [13, 14], but without requiring the author of the document to know CG. The interest of our approach is that if RDF, recommended by W3C, is widely adopted as a standard by the Web community, then a Web document author can both continue to use RDF annotations and draw benefit from the CG formalism, even without knowing himself the CG formalism.

The exploitation of RDF schemas by means of Conceptual Graphs seems more relevant in the context of a company or of a given community: this company or community can agree on the conceptual vocabulary used for expressing the metadata about their documents.

In the future, we plan to study a query language for RDF statements and the mapping to appropriate CG projections.

Acknowledgement

We thank Ahmed Amerkad who implemented the CG to RDF translator, Peter Eklund, Philippe Martin and Denis Kuntz for proof-reading the paper, the remaining errors being ours, and the ARC ESCRIRE for the partial funding of this work.

References

1. T. Berners-Lee. Metadata Architecture.
`http://www.w3.org/DesignIssues/Metadata.html`
2. T. Berners-Lee. Semantic Web Road map.
`http://www.w3.org/DesignIssues/Semantic.html`
3. T. Berners-Lee, D. Connolly, R. R. Swick. Semantic Web.
`http://www.w3.org/1999/04/WebData.html`
4. T. Bray, J. Paoli and C. M. Sperber-McQueen, Extensible Markup Language (XML) 1.0, W3C Recommendation, `http://www.w3.org/TR/REC-xml`, 1998

5. D. Brickley and R.V. Guha eds. Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 27 March 2000, http://www.w3.org/TR/rdf-schema
6. M. Chein, M. L. Mugnier. Conceptual Graphs: Fundamental Notions. Revue d'Intelligence Artificielle, vol. 6, n. 4, p. 365-406, 1992.
7. J. Clark. The XT Extensible Stylesheet Language (XSLT) engine. http://www.jclark.com.
8. D. Fensel, S. Decker, M. Erdmann, R. Studer. Ontobroker: Or How to Enable Intelligent Access to the WWW. In B. Gaines, M. Musen eds, Proc of the 11th Workshop on Knowledge Acquisition, Modeling and Management (KAW'98), Banff, Canada, April 18-23. http://ksi.cpsc.ucalgary.ca/KAW/KAW98/fensel1
9. F. Van Harmelen, D.Fensel. Practical Knowledge Representation for the Web. Proc. of IJCAI'99 Workshop on Intelligent Information Integration, Stockholm, Sweden, 1999. http://www.cs.vu.nl/ frankh/postscript/IJCAI99-III.html
10. O. Lassila 1998, Web Metadata: A Matter of Semantics. IEEE Internet Computing, July-August 1998, Vol. 2, No. 4.
11. O. Lassila and R. R. Swick eds. Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation 22 February 1999, http://www.w3.org/TR/REC-rdf-syntax.
12. S. Luke, L. Spector, D. Rager, J. Hendler. Ontology-based Web Agents. In Proc. of the First Int. Conference on Autonomous Agents, 1997.
13. P. Martin and P. Eklund. Embedding knowledge in web documents: CGs versus XML metadata languages. Proc. of the 7th Int. Conf. on Conceptual Graphs (ICCS'99), Springer-Verlag, August 1999.
14. P. Martin and P. Eklund. Embedding knowledge in web documents. Proc. of the 8th Int. World Wide Web Conf. (WWW8), p. 324-341, Elsevier, 1999.
15. A. Michard. XML : Langage et applications. Eyrolles, 1999.
16. F. Southey, Univ. of Guelf, Canada, http://dorian.cis.uoguelph.ca/CG/projects/notio, 1999
17. J. F. Sowa. Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Reading, 1984.
18. J. F. Sowa. Conceptual Graph Standard, http://www.bestweb.net/ sowa/cg/cgdpans.htm, 1999.
19. K. Tolle. The VRP RDF Parser, ICS Forth, Greece, http://www.ics.forth.gr/proj/isst/RDF

Appendix : the type system

This appendix presents (part of) the type system as it is implemented in the prototype. We also present (part of) the metamodel as it is implemented. These meta types are not intended to be instantiated in conceptual graphs, but only to be specialized in meta schema.

```
rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
rdfs='http://www.w3.org/TR/1999/PR-rdf-schema-19990303#'
concept type rdf:Thing
concept type rdfs:Literal   < rdf:Thing
concept type rdfs:Resource  < rdf:Thing
concept type rdfs:Container < rdfs:Resource
```

```
concept type rdf:Bag        < rdfs:Container
concept type rdf:Alt        < rdfs:Container
concept type rdf:Seq        < rdfs:Container
concept type rdf:Statement  < rdfs:Resource
relation type rdf:Property(rdfs:Resource, rdf:Thing)
relation type rdf:li(rdfs:Container, rdf:Thing) < rdf:Property

concept type rdfs:Class   < rdfs:Resource
concept type rdf:Property < rdfs:Resource
rel. type rdf:type(rdfs:Resource, rdfs:Class)     < rdf:Property
rel. type rdfs:subClassOf(rdfs:Class, rdfs:Class) < rdf:Property
rel.type rdfs:subPropertyOf(rdf:Property,rdf:Property)<rdf:Property
rel. type rdfs:ConstProperty(rdfs:Resource, rdf:Thing)<rdf:Property
rel. type rdfs:domain(rdf:Property, rdfs:Class)< rdfs:ConstProperty
rel. type rdfs:range(rdf:Property, rdf:Thing) < rdfs:ConstProperty
rel. type rdf:object(rdf:Statement, rdf:Thing)       < rdf:Property
rel. type rdf:subject(rdf:Statement, rdfs:Resource)  < rdf:Property
rel. type rdf:predicate(rdf:Statement, rdf:Property) < rdf:Property
```